Project 3: Correlation Attack

Fredrick Nilsson

December 6, 2023

Exercise 1

I found the following initial states:
K_1: [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1]
K_2: [0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1]
K_3: [1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0]
For the given output sequence: 10010000111011001011010110010010110101
10101110011010101010010000001100100110001110111011011110001000111111
11010101000001000010101110110000111100011100001010000110010011000000
01000110001110111101010

Exercise 2

Source code

```
1
2
3  pub fn exercise1() {
4      let num = include_str!("../task06.txt").chars().map(|c|
       c.to_digit(10).unwrap() as u8).collect::<Vec<u8>>();
5
6      // Primitive polynomials
7      let prim: Vec<Vec<u8>> = vec![
8          vec![1,0,1,1,0,0,1,1,0,1,0,1,1],
9          vec![1,0,1,0,1,1,0,0,1,1,0,1,0,1,0],
10         vec![1,1,0,0,1,0,0,1,0,1,0,0,1,1,0,1,0]];
11
12     // Generate de Bruijn sequences for the primitive
       polynomials
13     let seq: Vec<Vec<u8>> = prim.iter()
14         .map(|p| lfsr(p, vec![0; p.len()], 2_usize.pow(p.len
       () as u32))).collect();
15
16     // Find the position of the states with maximum
       correlation to the given number
17     let pos = seq.iter().map(|s| max_p(s, &num)).collect::<
       Vec<usize>>();
18
19     // Get the specific states of the sequence with the
       maximum correlation
20     let states = seq.iter().zip(pos.iter()).map(|(s, i)| s[*
       i..*i+num.len()].to_vec()).collect::<Vec<Vec<u8>>>();
21
22     // Confirm that the three sequences generate the given
       number
23     if check_seqs(num.clone(), &states){
24         println!("Found!");
25         println!("State␣1:␣{:?}", seq[0][pos[0]..pos[0]+prim
       [0].len()].to_vec());
```

```
26          println!("State 2: {:?}", seq[1][pos[1]..pos[1]+prim
    [1].len()].to_vec());
27          println!("State 3: {:?}", seq[2][pos[2]..pos[2]+prim
    [2].len()].to_vec());
28      } else {
29          println!("Not found!");
30      }
31  }
32
33  // Checks if the sequences generate the given number with
    majority vote
34  fn check_seqs(num: Vec<u8>, seq: &Vec<Vec<u8>>) -> bool{
35      num == seq[0].iter()
36          .zip(seq[1].iter())
37          .zip(seq[2].iter())
38          .map(|((x,y),z)| (*x+*y+*z) / 2)
39          .collect::<Vec<u8>>()
40  }
41
42  // Finds the position of the maximum correlation between the
     given sequence and the given number
43  fn max_p(seq: &Vec<u8>, num: &Vec<u8>) -> usize {
44      let mut dists: Vec<f32> = Vec::new();
45      for i in 0..(seq.len()-num.len()) {
46          let j = i + num.len();
47          let state = seq[i..j].to_vec();
48          let dist = distance(state, num.clone());
49          dists.push(dist);
50      }
51      let (pos, max) = dists.iter().enumerate().max_by(|(_, x)
    ,(_, y)| x.partial_cmp(y).unwrap()).unwrap();
52      println!("Found max: {}", max);
53      pos
54  }
55
56
57  // Generates a lfsr sequence of length len, starting with
    the state init and using prim as the primitive polynomial
```

```rust
fn lfsr(prim: &Vec<u8>, init: Vec<u8>, len: usize) -> Vec<u8> {
    let mut seq: Vec<u8> = init.clone();
    for _ in 0..len {
        let last = seq.as_slice()[seq.len()-prim.len()..].to_vec();
        if last[1..].to_vec() == vec![0_u8; prim.len()-1] {
            // special case of 0 state
            seq.push(if last[0]==1 {0} else {1});
        } else {
            // general case
            seq.push(and(last,prim.clone()).iter().sum::<u8>() % 2);
        }
    }
    seq
}

// Bitwise and of two vectors
fn and(a: Vec<u8>, b: Vec<u8>) -> Vec<u8> {
    let mut out: Vec<u8> = vec![0; a.len()];
    for i in 0..a.len() {
        out[i] = a[i] * b[i];
    }
    out
}

// Calculates the distance between two vectors
fn distance(a: Vec<u8>, b: Vec<u8>) -> f32 {
    let n = a.len() as f32;
    1.0 - hamming(a,b) as f32 / n as f32
}

// Calculates the hamming distance between two vectors
fn hamming(a: Vec<u8>, b: Vec<u8>) -> u8 {
    let mut out: u8 = 0;
    for i in 0..a.len() {
        out += (a[i] ^ b[i]) as u8;
```

```
93      }
94      out
95  }
```